Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut järjestelmät

2013

Juho Koskimäki

WCF-PALVELUN INTEGROINTI MOBIILISOVELLUKSEEN

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut järjestelmät

2013 | Sivumäärä 43

Ohjaaja: Yliopettaja Mika Luimula

Juho Koskimäki

WCF-PALVELUN INTEGROINTI MOBIILISOVELLUKSEEN

Windows Communication Foundation (WCF) on Microsoftin kehittämä rajapinta, jolla voidaan luoda www-sovelluspalveluita, joita voidaan käyttää monella tavalla jaetuissa sovelluksissa. Opinnäytetyössä tutkittiin WCF–rajapintaa hyödyntäviä www-sovelluspalveluita ja niiden integroimista Windows Phone -mobiilisovellukseen.

Teoriaosuudessa käsiteltiin www-sovelluspalveluihin liittyviä keskeisiä käsitteitä ja standardeja, joihin kuului mm. SOAP ja REST. Tarkemmin paneuduttiin WCF-palveluun ja siihen olennaisesti liittyviin asioihin, kuten päätepisteisiin, sidoksiin, sopimuksiin ja WCF-palveluiden tietoturvaan.

Työn käytännön osassa luotiin Windows Phone 7.5 –käyttöjärjestelmän mobiilipelille www-sovelluspalvelu, jolla tallennettiin pelissä saadut pisteet tietokantaan sekä haettiin tulokset käytettäväksi mobiilisovelluksessa. Sovelluspalvelun toteutuksessa käytettiin SOAP- ja REST-standardien mukaista toteutusta. Ohjelmointikielenä käytettiin Visual Basic –kieltä ja palvelu isännöitiin Internet Information Services 7.0 –alustalle.

Työn tuotoksena saatiin luotua toimiva www-sovelluspalvelu, joka käytti kahta eri päätepistettä liikenteeseen sovelluspalvelun ja mobiilisovelluksen välillä.

ASIASANAT:

WCF, www-sovelluspalvelu, Web Service, Windows Phone, Visual Basic

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology | Embedded systems

2013 | Total number of pages 43

Instructor: Mika Luimula, Principal Lecturer

Juho Koskimäki

INTEGRATING WCF SERVICE IN A MOBILE APPLICATION

Windows Communication Foundation (WCF) is a Microsoft framework for building web services which can be utilized in many ways in distributed applications. This thesis is about integration of WCF web service and Windows Phone application.

The theoretical part of this thesis deals with common concepts and standards of web services like SOAP and REST. In particular, it focuses on the WCF web services and their essential parts such as endpoints, bindings, contracts and WCF web services security.

The practical part of this thesis describes how a WCF web service for Windows Phone 7.5 -operating system was built. It was used to save game scores in a database and retrieve those scores from that database. The web service was implemented using SOAP and REST -standards. Visual Basic (VB) was used as the programming language and the service application was hosted on the Internet Information Services 7.0 platform.

The result of this thesis was a working web service which used two endpoints for transporting data between the web service application and the mobile application.

KEYWORDS:

WCF, web services, Windows Phone, Visual Basic

SISÄLTÖ

KÄYTETYT LYHENTEET

API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IIS	Internet Information Services
REST	(Representational State Transfer) Arkkitehtuurinen tyyli luoda sovelluspalvelu.
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UDDI	(Universal Description, Discovery, and Integration) Yleinen sovelluspalvelukirjasto
URI	Uniform Resource Identifier
WAS	Windows Process Activation Services
WCF	(Windows Communication Foundation) Microsoftin kehittämä rajapinta, jolla voidaan luoda sovelluspalveluita.
Web Service	Www-sovelluspalvelu
WSDL	Web Services Definition Language
WWW	World Wide Web
XML	Extensible Markup Language

1 JOHDANTO

Www-sovelluspalveluiden (Web Services) käyttö mobiilisovelluksissa tuo paljon uusia mahdollisuuksia sovellusten suunnitteluun ja toteutukseen. Jatkuvasti kasvava älypuhelinten osuus mobiilimarkkinoilla tuo sovelluksien suunnittelijoille sovelluspalveluiden käytön entistä enemmän ajankohtaisemmaksi. Myös internetyhteyksien kasvava nopeus ja 4G-tekniikan yleistyminen tuo uusia mahdollisuuksia käyttää palveluita osana sovelluksia. Www-sovelluspalvelut sekoitetaan usein verkkopalveluihin. Esimerkiksi Wikipedia on verkkopalvelu, joka tarjoaa samalla REST (Representational State Transfer) -pohjaisen web ce -rajapinnan, jossa käyttäjät voivat lukea ja muokata Wikipediaa verkkoselaimella[1]. Verkkopalvelulla viitataan siis ihmiselle tarkoitettuun internetissä sijaitsevaan palveluun, kun taas sovelluspalvelulla Web-teknologioilla tuotettuun palvelurajapintaan.

Opinnäytetyön teoriaosassa kerrotaan yleisesti www-sovelluspalveluista, -palveluiden yleisimmistä käsitteistä sekä standardeista. Lisäksi käsitellään yleisesti WCF (Windows Communication on) -rajapintaa hyödyntäviä sovelluspalveluita ja niihin liittyviä käsitteitä.

Käytännön osassa kerrotaan, kuinka toteutettiin Windows Phone 7.5 -käyttöjärjestelmän mobiilipelille www-sovelluspalvelu, jolla pelin tulokset tallennettiin ja haettiin tietokannasta. Työssä näytetään esimerkein, kuinka toteutettiin sovelluspalvelu ja miten se liitettiin pelisovellukseen. Luvussa neljä käsitellään verkkopalvelun toteutus Visual Studiolla ja luku viisi sisältää mobiilipelisovelluksen liittämisen sovelluspalveluun. Peli ja www-sovelluspalvelu ovat toteutettu Visual Basic (VB) -kielellä ja jokainen koodiesimerkki on näin ollen myös VB -kieltä. Sovelluspalvelun puolella työ on rajattu käsittelemään WCF-rajapinnassa toimivia sovelluspalveluita, jotka on toteutettu REST- tai SOAP (Simple Object Access Protocol) -tekniikoilla. Asiakaspuolella keskitytään Windows Phone 7.5 -käyttöjärjestelmään.

Työn tavoitteena on antaa lukijalle perustietoja www-sovelluspalveluista ja niihin liittyvistä aiheista. Lisäksi tavoitteena on saada luotua palvelu, joka pystytään yhdistämään Windows Phone 7.5 –mobiilisovellukseen.

2 WWW-SOVELLUSPALVELUIDEN STANDARDIT

Ennen www-sovelluspalveluja sovellukset niin työpöytä- kuin verkkosovellukset olivat hyvin eristäytyneitä toisistaan. Kahden eri alustoilla toimivan sovelluksen keskinäinen kommunikointi oli hyvin rajallista. Sovellusten toiminnallisuus oli kaikilta osin ohjelmoitu itse sovellukseen. Tämän takia sovelluksen päivitys vaati aina koko sovelluksen päivityksen, mikä taas johti kokonaiskustannusten kasvuun. Vaatimusten kasvaessa kehitettiin jaetut sovellukset, joissa sovelluksen toiminnallisuus on jaettu eri tasoihin. Yleisesti käytetään kolmea eri tasoa. Esitystasolla on käyttäjälle näytettävä toiminnallisuus eli käyttöliittymä. Keskitasolla määritetään business-logiikka, joka voi olla vaikka web service -palvelu, joka tarjoaa yhteyttä johonkin tietokantaan. Tietotasolla on itse tietokanta. Näin ollen web service -palvelua voidaan päivittää päivittämättä tietokantaa tai käyttöliittymää. [2]

Www-sovelluspalvelut pystytään toteuttamaan monella eri ohjelmointikielellä. Esimerkiksi suosituilla C#-, VB-, Java- ja PHP–ohjelmointikielillä on mahdollista toteuttaa sovelluspalvelu. Palveluiden tärkeimpiin hyötyihin kuuluukin se, ettei palvelun toteutuskielellä ole merkitystä, koska tietoa siirretään yleisiin standardeihin perustuvia protokollia käyttäen. Tämä myös tarkoittaa sitä, että oikein toteutettu sovelluspalvelu on käyttöjärjestelmäriippumaton.

2.1 Hypertext Transfer Protocol

Hypertext Transfer Protocol (HTTP) on tiedonsiirtoprotokolla, jolla yleisesti www-sovelluspalvelut siirtävät tietoa. HTTP käyttää porttia 80. Jos SSL (Secure Sockets Layer) salausta käytetään tiedonsiirrossa, siirtoprotokolla on HTTPS (Hypertext Transfer Protocol Secure). [3]

2.2 Extensible Markup Language

Extensible Markup Language (XML) on tiedon merkitsemistapa tai standardi, jota yleisesti käytetään verkkopalvelun ja asiakkaan välisessä tiedonsiirrossa. XML:ssä tieto järjestetään hierarkkisesti puurakenteeseen niin, että itse tieto on käärityt tunnisteisiin, jotka yleensä kertovat, mitä tietoa tunnisteet sisältää. Jokaisella aloitustunnisteella on siis oltava lopetustunniste. Näitä tunnisteita sanotaan merkkauksiksi ja tietoa merkkausdataksi. Koodiesimerkissä 1 näytetään yksinkertainen XML-tiedoston malli.

Koodiesimerkki 1 Yksinkertainen XML-esimerkki

```
<country>
    <name>Finland</name>
    <population>5420000</population>
</country>
<country>
    <name>Sweden</name>
    <population>9500000</population>
</country>
```

XML:ään liittyy myös nimitykset XML-skeema (engl. schema) ja XML-nimiavaruus (engl. namespace). Skeemoilla voidaan määritellä sallittu XML-muotoisen tiedoston rakenne. Nimiavaruuksia käytetään jaottelemaan nimeämiskäytäntöjä, mikä taas estää eri asioita tarkoittavien, mutta samannimisten merkkauksien sekoittumisen keskenään.

2.3 Www-sovelluspalveluiden määrittelykieli

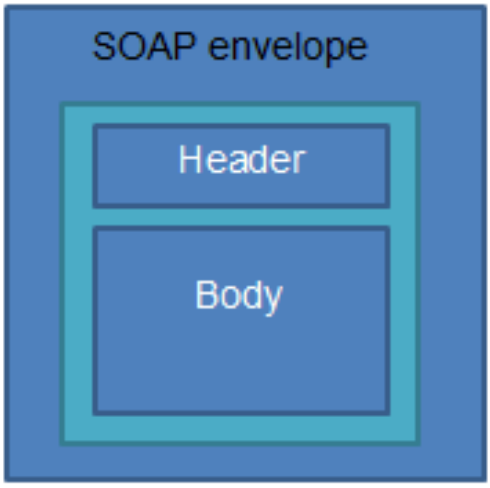
Web Services Definition Language (WSDL) on nimensä mukaan www-sovelluspalvelun määrittelykieli. WSDL-tiedosto on yleensä XML-muotoon kirjattu määritelmä kaikesta, mitä asiakassovelluksen tulee tietää verkkopalvelusta. WSDL liittyy SOAP-standardin mukaisiin sovelluspalveluihin. Se kertoo asiakkaana toimivalle sovellukselle mm. seuraavat tiedot palvelusta [2]:

- Määritykset

- Tietotyypit
- Viestit
- Sopimukset
- Porttityypit
- Sidokset.

2.4 Simple Object Access Protocol

Simple Object Access Protocol on protokolla, jolla XML-tyyppisiä viestejä siirretään internetin välityksellä[4]. SOAP perustuu Request-Response malliin. Protokollassa siirretään SOAP-viestejä palvelun ja asiakkaan välillä. Viestin rakenne on XML-muotoinen. Kuten kuvassa 1 näkyy, SOAP envelope on juuritason elementti SOAP-viestissä. Header ja Body ovat kirjeen lapsielementit.



Kuva 1 SOAP-kirje

SOAP-viestin otsikko(engl. header) -elementti ei ole pakollinen siirrettäessä SOAP-viestiä. Otsikossa lähetettävää tietoa ei ole erikseen määritelty, mutta yleisin tapa käyttää sitä on liittää siihen todennustietoa. Header-elementtejä voi olla viestissä useita. [5]

Body-elementissä välitetään itse viesti XML-muodossa. Body-elementin sisällä voi viestin lisäksi olla vikatietoa, joka siirretään Fault-tunnisteen sisällä. Vikatieto

ei ole pakollinen. Koodiesimerkissä 2 näytetään XML-muotoisen SOAP-viestin runko, myös XML-nimiavaruus ja koodaustyyli täytyy määritellä viestiin esimerkissä olevalla tavalla.

Koodiesimerkki 2 SOAP-viestin runko [6]

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Header>
    </soap:Header>
    <soap:Body>
        <soap:Fault>
        </soap:Fault>
    </soap:Body>
</soap:Envelope>
```

2.5 Representational State Transfer

Representational State Transfer on arkkitehtuurinen tyyli. REST-tyylillä luotua sovelluspalvelua kutsutaan usein RESTful web services nimellä. Arkkitehtuurissa palvelun resurssit paljastetaan suoraan osoitteella eli URIlla HTTP –protokollaa käyttäen. Jokaisella resurssilla on oma osoite. REST-tyylin toteutetussa sovelluksessa käytetään HTTP–protokollan tarjoamia perusmetodeita kuten GET, PUT ja DELETE. WWW (World Wide Web) on hyvä esimerkki REST-tyylin toteutuksesta. Selain tekee kyselyn www–osoitteen perusteella. Osoite palauttaa kyseisen sivun muodossa, jonka selain osaa renderöidä näkyväksi sivuksi. [3]

Tiedonsiirtomuoto RESTful sovelluspalvelussa voi olla XML-muodon lisäksi myös JSON(JavaScript Object Notation). JSON on erityisesti JavaScriptiä sovelluksissa käyttävien web-sivujen usein hyödyntämä muoto.

RESTful sovelluspalveluiden suurin hyöty saadaan suuria tietoja siirtäessä, sillä SOAP-standardiin verrattaessa viestin koko on pienempi. Tämä johtuu siitä, että SOAP-viesti tarvitsee SOAP-kirjeen, joka kasvattaa jokaisen viestin kokoa.

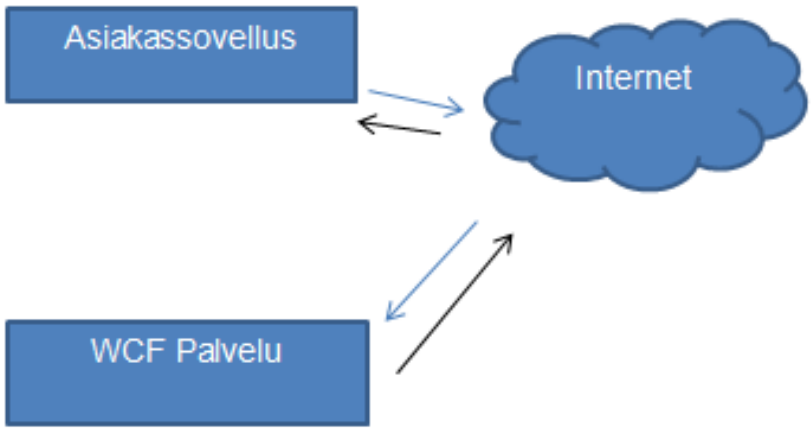
2.6 UDDI

The Universal Description, Discovery and Integration (UDDI) –protokolla on hyväksytty OASIS-standardi ja avain jäsen Web Service –pinossa. Se määrittelee standardi tyylin palvelukeskeiseen arkkitehtuuriin perustuvien verkkosovelluskomponenttien julkaisuun ja löytöön. [7]

UDDI on siis oikeastaan tietopankki, johon voidaan lisätä Web ce -palveluja muiden asiakkaiden käyttöön. UDDIsta yritykset tai yksityiset sovelluskehittäjät voivat etsiä tarpeisiinsa sopivia valmiita palveluja käytettäväksi omissa sovelluksissaan.

3 WINDOWS COMMUNICATION FOUNDATION

WCF on Microsoftin luoma kehys, jolla voidaan luoda palvelukeskeiseen arkkitehtuuriin perustuvia sovelluksia. Palvelukeskeinen arkkitehtuuri (SOA) on suunnittelutapa, jolla pyritään luomaan avoimien standardien rajapintoja hyödyntäviä palveluita. Tällaisia rajapintoja ovat muun muassa HTTP, SMTP ja TCP. Käytännössä tämä tarkoittaa sitä, että palvelusovelluksen integraatio eri teknologioita käyttävien järjestelmien välillä helpottuu. Yksinkertaisimmillaan voidaan sanoa, että WCF on joukko API (Application Programming Interface) -rajapintoja, joilla voidaan luoda järjestelmä, joka lähettää ja vastaanottaa viestejä palvelun ja asiakkaan välillä. Asiakassovelluksena voidaan käyttää toteutuksesta riippuen muun muassa verkkoselainta, toista www-sovelluspalvelua tai mobiilisovellusta (kuva 2).

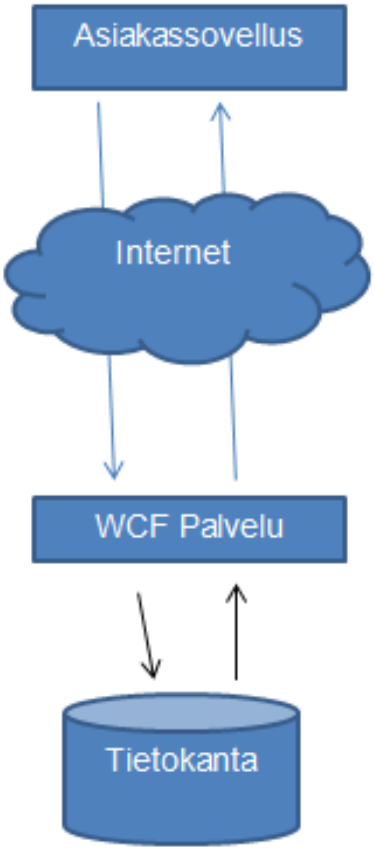


Kuva 2 WCF-palvelun toimintamalli

WCF-sovelluspalvelu tarjoaa asiakkaalle määriteltyjä toimintoja, joita asiakas pystyy käyttämään. Viestien lähetys tapahtuu asynkronisesti (ei-reaaliaikaisesti), mikä tarkoittaa, etteivät palvelun osapuolet ole ajasta tai paikasta riippuvaisia. Viestit voivat olla hyvin yksinkertaisia, kuten yksittäisiä merkkejä tai hyvin monimutkaista binääristä tietovirtaa. [8]

Liikenne palvelun ja asiakkaan välillä määritellään päätepisteillä. Päätepisteessä määrätään palvelun osoite, sidos ja sopimus. Osoite kertoo palvelun fyysisen paikan internetissä, sidokset määräävät millä tavalla viesti välitetään ja sopimus kertoo mitä toimintoja palvelulla on. Asiakassovellus ottaa yhteyden WCF-sovellukseen juuri päätepisteiden avulla. Asiakassovellus luo WCF-sovelluksen tarjoamaa päätepistettä vastaavan yhteyskäytännön. Sovelluksen voi toteuttaa käyttämällä esimerkiksi REST- tai SOAP-tekniikoita. [8]

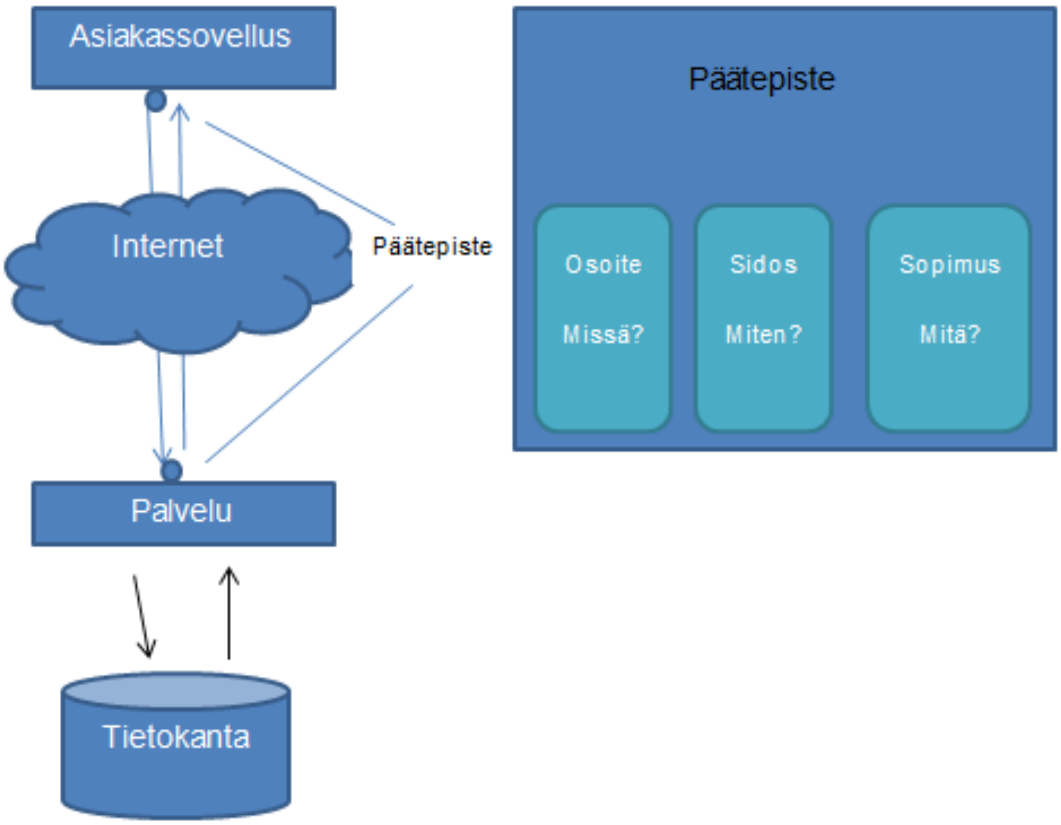
Kuvassa 3 esitetään hyvin yksinkertainen WCF-palvelu. Palvelu tarjoaa toiminnon, joka hakee tietokannasta päivän uutiset. Asiakas kutsuu www-sovelluspalvelun tarjoamaa toimintoa, joka palauttaa uutiset. Toiminto kutsuu tietokannan proseduuria, joka palauttaa uutiset sovelluspalvelulle, minkä jälkeen palvelu palauttaa uutiset asiakassovellukselle.



Kuva 3 WCF-esimerkki

3.1 Päätepisteet

WCF-palvelun päätepisteet määrittävät miten, missä ja mitä palvelua asiakas pystyy käyttämään. Liikenteen mahdollistamiseksi asiakkaan ja palvelun välillä on tärkeää, että päätepisteet ovat konfiguroitu vastaamaan toisiaan (kuva 4). [8]



Kuva 4 Päätepiste-esimerkki

Osoite

Päätepisteessä määritettävä osoite kertoo palvelua käyttävälle asiakkaalle missä palvelu fyysisesti sijaitsee. Esimerkiksi kehitysympäristössä osoite voisi olla ”service”, ja se kerrottaisiin päätepisteen määrittelyssä koodiesimerkin 3 tavalla.

Koodiesimerkki 3 Osoitteen määritys päätepisteelle

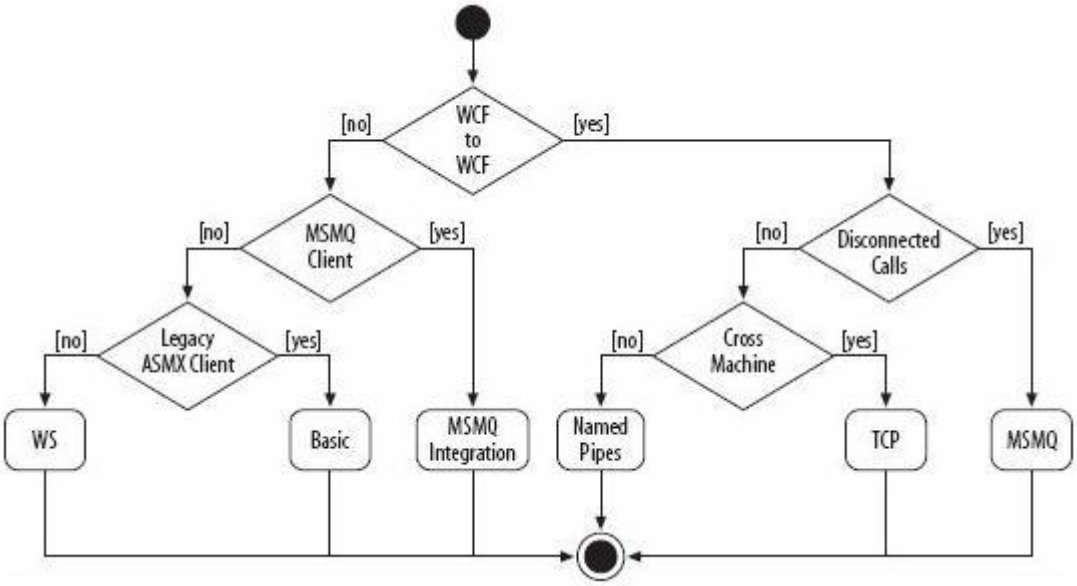
```
1   <endpoint address="/service" binding="webHttpBinding" contract="IService">
```

Sidokset

Sidoksien tarkoitus on kertoa millä yhteyskäytännöllä liikenne päätepisteellä mahdollistetaan. Jokaisella päätepisteellä täytyy olla vähintään yksi sidos. Määriteltäessä sidoksia täytyy ottaa huomioon, mikä sidostyyppi sopii parhaiten juuri sen hetkiseen palveluun. Yksittäistä oikeaa valintaa ei aina ole, joten suunnittelussa kannattaa käyttää aikaa oikean sidostyypin valintaan. WCF tarjoaa 12 eri sidostyyppiä, joista jokaisella on oma tarkoituksensa. Palvelun toteutustekniikka riippuu myös siitä minkä sidostyypin valitsee. Taulukossa 1 on esitelty lyhyesti tärkeimmät sidostyypit ja niiden selitys. Kuvassa 5 on esitetty kuva oikean sidostyypin valinnasta.

Taulukko 1 Sidostyypit [9]

Sidostyyppi	Selitys
BasicHttpBinding	A binding that is suitable for communicating with WS-Basic Profile conformant Web services, for example, ASP.NET Web services (ASMX)-based services. This binding uses HTTP as the transport and text/XML as the default message encoding
WsHttpBinding	A secure and interoperable binding that is suitable for non-duplex service contracts.
WsDualHttpBinding	A secure and interoperable binding that is suitable for duplex service contracts or communication through SOAP intermediaries.
WsFederationHttpBinding	A secure and interoperable binding that supports the WS-Federation protocol that enables organizations that are in a federation to efficiently authenticate and authorize users.
NetNamedPipeBinding	A secure, reliable, optimized binding that is suitable for on-machine communication between WCF applications.
NetTcpBinding	A secure and optimized binding suitable for cross-machine communication between WCF applications.
NetPeerTcpBinding	A binding that enables secure, multiple machine communication.
NetMsmqBinding	A queued binding that is suitable for cross-machine communication between WCF applications.
MsmqIntegrationBinding	A binding that is suitable for cross-machine communication between a WCF application and existing Message Queuing applications.
WebHttpBinding	A binding used to configure endpoints for WCF services that are exposed through HTTP requests instead of SOAP messages.



Kuva 5 Oikean sidostyypin valinta [10]

Päätepisteen sopimus

Päätepisteessä määritetyssä sopimusattribuutissa kerrotaan päätepisteelle, mitä palvelusopimusta päätepiste käyttää. Palvelusopimus sisältää kaikki palvelun asiakkaalle tarjoamat toiminnot. Sopimus voi sisältää yhden tai useita eri toimintoja. Toimintoja kutsutaan operaatiosopimuksiksi. <ServiceContract()> -tunniste määrittää palvelusopimuksen. Palvelun toiminnan kannalta palvelusopimus on pakollinen. Se voidaan määritellä joko luokassa kuten koodiesimerkissä 5 tai liittymässä (engl. interface) (koodiesimerkki 4). [11]

Koodiesimerkki 4 Palvelusopimus liittymässä

```
1  <ServiceContract()> _
2  Public Interface IService
3
4      <OperationContract()> _
5      Function GetData(ByVal value As Integer) As String
6
7  End Interface
```

Koodiesimerkki 5 Palvelusopimus luokassa

```
1  <ServiceContract()> _
2  Public Class IService
3
4      <OperationContract()> _
5      Function GetData(ByVal value As Integer) As String
6
7  End Class
```

Määritys

Palvelua luotaessa päätepisteet voidaan määritellä kahdella tavalla. Ensimmäinen tapa on määritellä päätepisteet palvelun konfiguraatiotiedostossa. Koodiesimerkissä 6 esitetään kuinka konfiguraatiotiedostoon lisätään päätepiste. Päätepisteen attribuutteina on osoite, sidos ja sopimus. Osoiteattribuutin ollessa tyhjä palvelu käyttää oletusosoitetta. Toinen mahdollisuus päätepisteen määrittämiseen on tehdä se dynaamisesti ilman konfiguraatiotiedostomuutosta. Koodiesimerkissä 7 näytetään, kuinka päätepiste lisätään palvelulle dynaamisesti. [12]

Koodiesimerkki 6 Päätepisteen lisäys konfiguraatiotiedostoon

```
<system.serviceModel>
    <services>
      <service name="Service" behaviorConfiguration="ServiceBehavior">
        <!-- Palvelun päätepisteet -->
        <endpoint address="" binding="webHttpBinding" contract="IService"/>
      </service>
    </services>
  </system.serviceModel>

```

Koodiesimerkki 7 Päätepisteen lisäys dynaamisesti

```
' Osoite
Dim osoite As String =  "http://localhost/Service"
' Sidos
Dim sidos As New BasicHttpBinding()

  Using host As New ServiceHost(GetType(Service))
    With host
        .AddServiceEndpoint(GetType(IService), _
                            sidos, _
                            osoite)
```

3.2 Operaatiosopimukset

Kaikki asiakkaan käytettävissä olevat WCF-palvelun toiminnot merkitään operaatiosopimuksilla. Tämä tarkoittaa, että <OperationContract()> -tunniste täytyy löytyä jokaisesta palvelusopimuksen funktiosta. Funktioilla on aina joko yksi tai useampi parametri sekä palautusarvo.

3.3 Sarjoitus

Kun luodaan verkkopalvelua, tietotyyppien valinta on tärkeää. Kaikki toimintojen käyttämät parametrit ja palautusarvot täytyy olla sarjoittuvia. Näihin kuuluu kaikki System-nimiavaruuteen kuuluvat tietotyypit esimerkiksi integer ja string. Sarjoittuvalla tarkoitetaan objektia, joka pystytään muuttamaan bittivirraksi. Tarkoituksena on pystyä säilyttämään, ei vain tieto, mutta myös tietotyyppi. Näin ollen bittivirta pystytään muuttamaan takaisin objektiksi. Koska www-sovelluspalvelua käytetään yleensä verkkoprotokollien yli, ainoa tapa siirtää tietoa on bittivirtoja

käyttämällä. Tämän takia sarjoitus on pakollinen, mutta se suoritetaan normaaleille tietotyypeille automaattisesti.

3.4 Tietotyyppisopimukset

Joskus verkkopalveluun halutaan siirtää monimutkaista tietoa, jonka tietotyyppi ei ole sarjoittuva. Tähän on kehitetty tietotyyppisopimukset. Näillä sopimuksilla pystytään sarjoittamaan omia tietotyyppejä. WCF käyttää oletuksena datacontract serializer -moottoria omien tietotyyppien sarjoitukseen ja sarjoituksen purkuun. Palvelun sarjoitusmoottori on mahdollista määrittää myös käyttämään vanhempaa XML serializer -moottoria. Tällöin kuitenkin pitää muistaa asiakassovelluksen puolella suorittaa sarjoitus samaa moottoria käyttäen.

3.5 Tietoturva

Www-sovelluspalvelun suunnittelussa on tärkeää huomioida tietoturva. Tietoa siirretään internetin läpi, jolloin on aina vaara, että joku kaappaa tiedon ennen kuin se pääsee päämääräänsä. Kun lähetetään arkaluonteista dataa kuten pankkitietoja, pitää viesti olla salattua. Tämä tarkoittaa, ettei viestin sisältö paljastu, vaikka sen joku saisikin kaapattua ennen kuin se saapuu vastaanottajalle. On myös tärkeää huomioida, että asiakas ei aina ole se ketä se väittää olevansa. Tietoturva www-sovelluspalveluissa voidaan jakaa kolmeen osioon: asiakkaan tunnistukseen, viestin salaukseen ja liikenteen salaukseen. Sovelluspalveluun valittu päätepisteen sidos määrää, minkälaista turvallisuusprotokollaa on mahdollista käyttää.

Asiakkaan tunnistus

Sovelluspalvelua suunniteltaessa on tärkeää pystyä tunnistamaan asiakas. Ei haluta, että pankkitietoja tarjoava palvelu luovuttaa tietoja väärille asiakkaille.

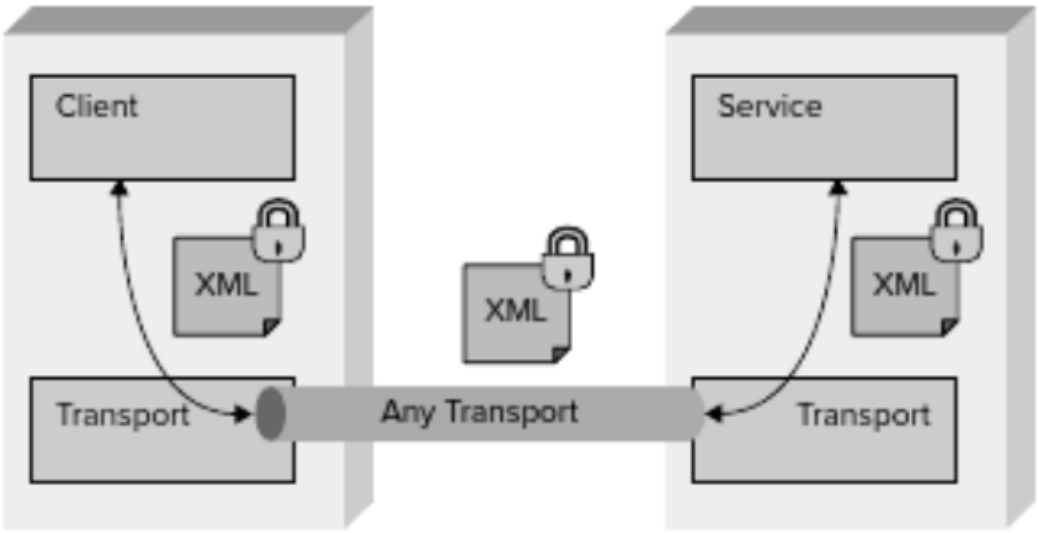
Asiakas voi olla esimerkiksi loppukäyttäjä tai yhtä hyvin toinen verkkopalvelu. Asiakkaan tunnistuksessa pitää muistaa ottaa huomioon seuraavat kaksi asiaa:

- Onko asiakas se joka väittää olevansa?
- Miten varmistun siitä, että asiakas on kuka väittää?

Asiakkaan tunnistamiseksi voidaan kehittää useita eri menetelmiä kuten mm. käyttäjänimi ja salasanaparit, sertifikaatit tai Kerberos-liput. Tärkeää näissä kaikissa on kuitenkin se, että sovelluspalvelu pystyy varmistamaan tarjotun tunnistustiedon aitouden. Tietenkin jos tunnistus epäonnistuu, pääsy palveluun on estettävä. [8]

Viestin salaus

Koska viestien lähetys www-sovelluspalvelun ja asiakkaan välillä tapahtuu verkkoprotokollia hyödyntäen, viestit ovat yleensä selkokielisenä luettavana koko matkan lähteen ja päämäärän välillä. Jos lähetettävä tieto on arkaluontoista, on tärkeää, että viestit lähetetään muodossa, jossa niitä ei voida lukea. Kuvassa 6 näytetään kuinka viesti lähetetään asiakkaan ja palvelun välillä salattuna. Asiakas salaa tiedot ja vain palvelu tietää kuinka salaus puretaan. Tällä tavoin tietoliikenne voidaan suorittaa käyttämällä mitä tahansa tietoliikenneprotokollaa varmistuen siitä, että tieto ei vaarannu asiakkaan ja palvelun välillä. [8]



Kuva 6 Viestin salaus asiakkaan ja palvelun välillä [8]

Liikenteen salaus

Liikenteen salauksella tarkoitetaan salattujen tietoliikenneprotokollien käyttöä. Kun käytetään pelkkää liikenteen salausta, viesti itsessään voidaan lähettää salaamattomana. Tätä tapaa käytettäessä täytyy kuitenkin muistaa, että jos tieto kiertää sellaisen sovelluspalvelun kautta, mikä ei käytä suojattua tietoliikenneprotokollaa, on viesti tällöin turvaton. Liikenteen salausta pitäisikin käyttää vain end-to-end-tapauksissa, mikä tarkoittaa, että tieto lähetetään suoraan vastaanottajalle, jolloin tietoturva säilytetään. [8]

3.6 Isännöinti

WCF–palvelu ei voi olla toiminnassa ns. yksinään. Tarvitaan aina joku paikka, jossa palvelu toimii. Voidaan sanoa, että isäntä on alusta jonka päällä palvelu pyörii. Isännöinti voidaan hoitaa monella tavalla. Tärkeimpiä isännöintialustoja ovat:

- Self-hosting (itseisännöinti)

- Windows Services
- IIS
- IIS7/Windows Process Activation Service (WAS).

Itseisännöinti

Yksi WCF-palvelun isännöintialusta on itseisännöinti. Tämä tarkoittaa, että palvelun isäntänä toimii sitä käyttävä sovellus. Kaikki .NET-sovellukset, jotka pystyvät luomaan ServiceHost ilmentymän, voivat toimia isäntinä WCF-palvelulle. Itseisännöinnissä huomioitavaa on se, että palvelu ei ole viestikeskeinen, mikä tarkoittaa, että sovelluksen täytyy käynnistää ja sammuttaa palvelu itse. Itseisännöintiä käyttämällä on HTTP ainoa mahdollinen tiedonsiirtoprotokolla.

Itseisännöintiä käytetään usein palvelun kehitysvaiheessa, jolloin testaus helpottuu ja viestien sisältöä on helpompi seurata. [13]

Windows Services

Toinen tapa isännöidä www-sovelluspalvelu on käyttää Windows Servicesiä. Tällä tavoin isännöity palvelu toimii minkä tahansa Windows–käyttöjärjestelmän päällä. Windows Service on siis käyttöjärjestelmässä ajettava prosessi. Palvelu täytyy asentaa käyttöjärjestelmään käyttämällä .NET-kehyksen Instalutil.exe työkalua. Palvelu voidaan asettaa käynnistymään aina kun Windows käynnistetään. Samaan tapaan kuin itseisännöinnissä Windows Service ei myöskään ole viestikeskeinen eikä muuta kuin HTTP-protokollaa voida käyttää. [14]

Internet Information Services

Isännöinniksi voidaan valita myös IIS (Internet Information Services). IIS on joustava, turvallinen ja hallittava Web-palvelin minkä tahansa sisällön isännöintiin verkossa[15]. Käyttämällä IIS-palvelinta WCF-palvelu pystyy hyödyntämään palvelimen tarjoamia hyötyjä joita ovat mm. prosessien kierrätys, joutosammutus, prosessien tilan valvonta ja viestikeskeinen aktivointi. Vanhemmissa IIS versioissa ainoa mahdollinen tietoliikenneprotokolla on HTTP. IIS 7 -versiosta alkaen muutkin WCF-palvelun tukemat protokollat ovat käytettävissä. [16]

IIS7/Windows Process Activation Service

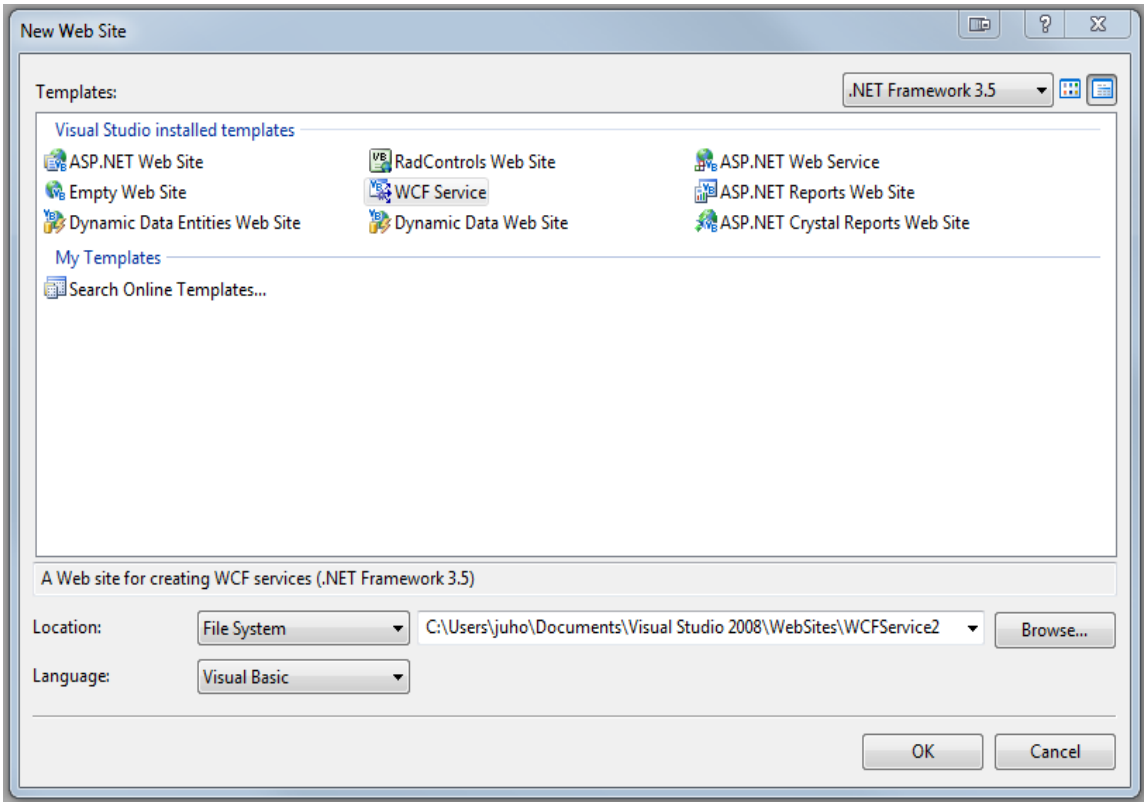
Windows Process Activation Service (WAS) on IIS7 versioon mukaan tullut isännöintitapa. Se sisältää kaikki aiempien versioiden hyödyt. WASia voidaan käyttää myös ilman, että kaikkia IISin osia on asennettu. Tämän lisäksi WAS tuo muita WCF:n tukemia tiedonsiirtoprotokollia käytettäviksi. Näitä ovat mm. TCP, MSMQ ja named pipes.[17]

4 WCF-PALVELUN LUONTI

Tässä ja seuraavissa luvuissa käydään läpi yksinkertaisen WCF-palvelun luonti Visual Studio 2008:lla. Arkkitehtuurina toimii .NET Framework 3.5. Palvelu toteutettiin käyttämällä SOAP- ja REST-tekniikoita samassa www-sovelluspalvelussa. Palvelu isännöidään IIS7-palvelimella. Palvelua käytettiin mobiilipelin verkkotulosten tallentamiseen tietokantaan.

4.1 Mallipohjat

Palvelun luonti aloitettiin Visual Studion mallipohjia hyväksikäyttäen. Visual Studio 2008 tarjoaa WCF Service -nimisen mallipohjan, josta on hyvä muokata omiin tarpeisiin sopiva sovelluspalvelu. Mallipohjan luominen aloitettiin luomalla uusi verkkosivu ja valitsemalla WCF Service –mallipohja (kuva 7).

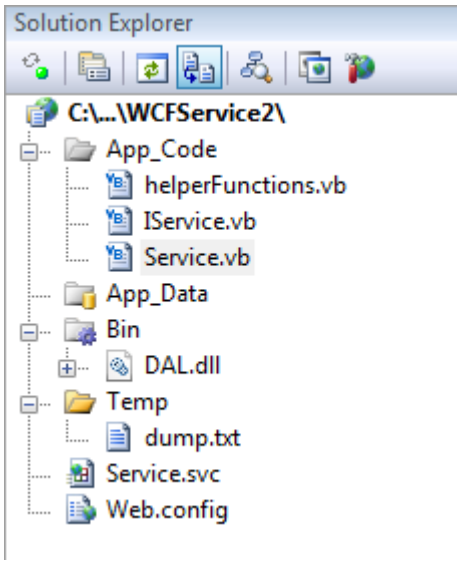


Kuva 7 Mallipohjan valinta

Valinnan jälkeen Visual Studio loi automaattisesti www-sovelluspalvelun, jossa oli kaikki palvelun tarvitsemat tiedostot ja oletusasetuksilla muokattu web.config.

4.2 Tiedostot

Www-sovelluspalvelun tarvittavat tiedostot luodaan automaattisesti oletusasetuksilla mallipohjan valinnan jälkeen. Kuvassa 8 on sovelluspalvelussa käytetyt tiedostot. Verkkopalvelun toiminnan kannalta tarvittavat tiedostot ovat App_Code-kansiossa sijaitsevat IService.vb ja Service.vb sekä projektin juuressa sijaitsevat Service.svc ja Web.config. Taulukossa 2 on esitetty käytettyjen tiedostojen selitykset.



Kuva 8 WCF-palvelun tiedostot

Taulukko 2 Tiedostojen selitykset

Tiedostonimi	Selitys
helperFunctions.vb	Luokkaan kirjataan testauksessa hyödynnettäviä funktioita. Täältä löytyy esimerkiksi funktio, joka kirjoittaa haluttua tietoa dump.txt-tiedostoon.
IService.vb	Tiedostoon määritellään liittymät ja sopimukset.
Service.vb	Tiedosto määrittää luokan, johon kaikki toiminnallisuus kirjataan. Tämä luokka toteuttaa (engl. implements) kaikki liittymät (engl. interface) ja näin ollen kaikki liittymien esittelemät funktiot löytyvät toiminnallisuuksineen täältä.
DAL.dll	Kirjastoa käytetään yhteytenä tietokantaan.
dump.txt	Tiedostoa käytetään testaukseen. Tänne voidaan kirjoittaa helperFunctions.vb-luokan dump–funktiolla. Tämä on hyvä tapa selvittää esimerkiksi tietokannasta tulevaa dataa tai muuta tietoa, josta testauksessa on hyötyä.
Service.svc	Tiedostossa on tieto palvelun nimestä, koodikielestä ja palvelun toiminnallisuuden sijainnista.
Web.config	XML-muotoinen tiedosto, jossa kaikki palvelua koskevat asetukset ovat määriteltynä.

4.3 Päätepisteet

Palvelulle luodaan päätepisteet tarpeiden mukaan. Ylimääräiset päätepisteet kannattaa poistaa, ettei mahdollisille hakkereille anneta mahdollisuuksia hyödyntää www-sovelluspalvelua niiden avulla. Yleensä päätepisteet määritetään web.config konfiguraatiotiedostoon. Päätepisteet päätettiin luoda käyttäen molempia REST- ja SOAP-tekniikoita. Palvelu olisi voitu toteuttaa myös käyttämällä vain toista tekniikkaa, mutta molemmat tekniikat valittiin, jotta saatiin parempi ymmärrys eri tekniikoista ja niiden eroista.

Kuten jo aiemmin todettiin päätepisteet tarvitsevat toimiakseen osoitteen, sidoksen ja sopimuksen. REST-päätepisteen käyttämäksi osoitteeksi valittiin kuvaava /rest. Tämä tarkoittaa, että sopimuksen julkistamat toiminnot löytyvät http://esimerkkiurl.net/rest/ paikasta. Koska käytössä on REST-tyyliin toteutettu päätepiste, asiakas pääsee käsiksi toimintoihin käyttämällä HttpWebRequest kutsua. Sopimuksen nimeksi valittiin IRest. Sidokseksi valittiin webHttpBinding, koska tämä sidos on ainoa REST tekniikan sidos. REST päätepiste tarvitsee myös käyttäytymismäärityksen, jonka määrityksen voi katsoa koodiesimerkistä 8.

SOAP tekniikan päätepiste asetettiin käyttämään /soap-osoitetta. Sidoksen valinnassa ainoa mahdollisuus oli käyttää basicHttpBinding–sidosta, koska tämä oli ainoa Windows Phone –käyttöjärjestelmän tukema sidos. Mobiilisovellus pystyy käyttämään SOAP-päätepistettä lisäämällä palvelureferenssin sovellusprojektiin ja luomalla asiakasyhteyden sitä kautta.

Koodiesimerkki 8 Päätepisteet Web.Config–tiedostossa

```
<system.serviceModel>
    <services>
        <service name="Service" behaviorConfiguration="ServiceBehaviour">
            <!-- Service Endpoints -->
            <endpoint address="/rest" binding="webHttpBinding" behaviorConfiguration="web" contract="IRest"/>
            <endpoint address="/soap" binding="basicHttpBinding" contract="ISoap"/>
        </service>
    </services>
    <behaviors>
        <endpointBehaviors>
            <behavior name="web">
                <webHttp/>
            </behavior>
        </endpointBehaviors>
        <serviceBehaviors>
            <behavior name="ServiceBehaviour">
                <serviceMetadata httpGetEnabled="true"/>
                <serviceDebug includeExceptionDetailInFaults="false"/>
            </behavior>
        </serviceBehaviors>
    </behaviors>
</system.serviceModel>
```

4.4 Sopimukset

Kaikki sopimusmääritykset tehdään aina omaan tiedostoonsa. Tässä tapauksessa tiedoston nimi on IService.vb. Sovelluspalvelulle luotiin kaksi palvelusopimusta, molemmille päätepisteille omansa. Tämä ei ole pakollista, mutta selkeyttää toteutusta. Koodiesimerkissä 9 on IService.vb tiedoston toteutus. Kuten jo kerrottiin, luotiin palvelulle kaksi palvelusopimusta, nämä löytyvät riveiltä 4 ja 12. SOAP-liittymä sisältää tuloksen tallennuksen funktion. Tässä huomioitavaa on funktion parametrina oleva Tulos-tietotyyppi, koska se ei ole normaali sarjoittuva muoto, vaan itse luotu. Tällöin tietotyyppisopimus täytyi määritellä. Määrittely alkaa riviltä 24. Tulosinstanssi sisältää tuloksentekijän nimen sekä pisteet, jotka molemmat ovat siis tietotyyppisopimuksen tietojäseniä. Nimi on string-muotoa ja Pisteet integer-muotoa.

REST-päätepisteen funktiot täytyy määritellä REST-funktioiksi WebGet- tai WebInvoke-attribuuteilla. WebGet-attribuutille annetaan parametriksi URI-osoite määrittelemällä UriTemplate. Oletusmuotona käytetään XML-muotoa, joten sitä ei tarvitse erikseen määrittää. Jos oltaisi haluttu käyttää JSON-muotoa, olisi

pitänyt UriTemplate-parametrin lisäksi määrittää ResponseFormat:=WebMessageFormat.Json.

Koodiesimerkki 9 IService.vb-tiedosto

```
Imports System.Data
Imports System.ServiceModel.Web

<ServiceContract()> _
Public Interface ISoap

    <OperationContract()> _
    Function TallennaTulos(ByVal tulos As Tulos) As Boolean

End Interface

<ServiceContract()> _
Public Interface IRest
    <WebGet(UriTemplate:="/haetop10/")> _
    <OperationContract()> _
    Function HaeTop10Tulokset() As DataTable

    <WebGet(UriTemplate:="/haesija/{pisteet}")> _
    <OperationContract()> _
    Function haeTulosSijoitus(ByVal pisteet As String) As Integer

End Interface

<DataContract(Namespace:="")> _
Public Class Tulos
    Private strTunnus As String
    Private intPistemaara As Integer

    <DataMember()> _
Public Property Pisteet() As Integer
        Get
            Return Me.intPistemaara
        End Get
        Set(ByVal value As Integer)
            Me.intPistemaara = value
        End Set
    End Property

    <DataMember()> _
    Public Property Tunnus() As String
        Get
            Return Me.strTunnus
        End Get
        Set(ByVal value As String)
            Me.strTunnus = value
        End Set
    End Property

End Class
```

4.5 Palvelu

Palveluluokassa määriteltiin itse funktioiden toiminnallisuus. Luokka implementoi molemmat liittymät, jolloin kaikki liittymäfunktiot täytyy löytyä myös palveluluokasta. Funktiot kutsuvat tietokantapalvelimelle luotuja proseduureja, jotka toteuttavat halutun toiminnallisuuden. Toiminnallisuudet on käärityy try catch -lohkoon, jolloin virheen sattuessa saadaan kaapattua virheilmoitus. Virheilmoitus kirjoitetaan helperFunctions.Dump()-funktiolla dump.txt-tiedostoon. Tietokantayhteys on toteutettu DAL.dll-kirjastolla, johon on myös luotu erilaisia proseduurin kutsumisfunktioita. Näitä ei käsitellä tässä enempää. Koodiesimerkissä 10 esitetään palveluluokan toteutus.

Koodiesimerkki 10 Palveluluokan toteutus

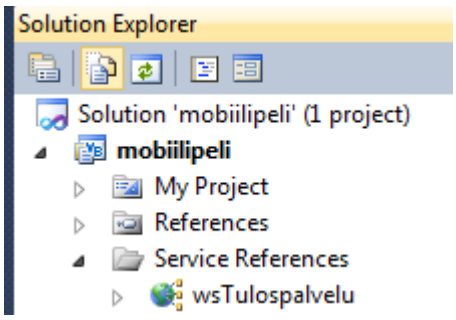
```
1  Imports System.Data
2  Imports DAL
3
4  Public Class Service
5      Implements ISoap, IRest
6
7      Private Shared objSQLXMan _
8      As New  _
9      ManSQL(ConfigurationManager.ConnectionStrings("ConnectionString").ToString)
10
11     Public Sub New()
12     End Sub
13
14     Public Function TallennaTulos(ByVal tulos As Tulos) _
15     As Boolean Implements ISoap.TallennaTulos
16         Try
17
18             Dim strTunnus As String = tulos.Tunnus.ToString
19             Dim intPistemaara As Integer = Integer.Parse(tulos.Pisteet)
20             objSQLXMan.ExecuteNonQuery("", "ws_tallenna_tulos", _
21                                        strTunnus, intPistemaara)
22             Return True
23         Catch ex As Exception
24             helperFunctions.Dump(ex.ToString)
25             Return False
26         End Try
27     End Function
28
29     Public Function HaeTop10Tulokset() _
30     As DataTable Implements IRest.HaeTop10Tulokset
31         Try
32             Dim ds As DataSet = _
33             objSQLXMan.ExecuteDataset("", "ws_hae_top10tulokset")
34             Dim datatable As DataTable = ds.Tables(0)
35             Return datatable
36         Catch ex As Exception
37             helperFunctions.Dump(ex.ToString)
38             Return Nothing
39         End Try
40     End Function
41
42     Public Function haeTulosSijoitus(ByVal intPisteet As String) _
43     As Integer Implements IRest.haeTulosSijoitus
44         Try
45             Return objSQLXMan.ExecuteScalar("", "ws_hae_tulos_sijoitus", _
46                                             intPisteet)
47         Catch ex As Exception
48             helperFunctions.Dump(ex.ToString)
49         End Try
50
51     End Function
52
53 End Class
```

5 WINDOWS PHONE JA WCF

Luvussa 4 käsiteltiin www-sovelluspalvelun toteutus mobiilipelin tulosten tallennukselle ja luvulle tietokannasta. Seuraavassa luvussa sovelluspalvelu otetaan käyttöön Windows Phone 7.5 –käyttöjärjestelmän mobiilisovelluksessa. Tallennus ja luku toteutettiin käyttämällä HttpWebRequest instanssia REST-funktioihin ja SoapClient-instanssia SOAP-funktioihin.

5.1 Sovelluspalvelun lisäys projektiin

SOAP päätepistettä käytettäessä on pakko lisätä sovelluspalvelu projektille. Www-sovelluspalvelu lisättiin Service references-kansion päältä painamalla Add Service reference, jonka jälkeen syötettiin osoite viittaamaan sovelluspalvelun osoitteeseen. Visual Studio latasi sovelluspalvelusta tarvittavat tiedot projektille. Palvelu tulee näkyviin onnistuneen lisäyksen jälkeen Service References-kansioon (kuva 9). Palvelun alta voi katsoa muun muassa WSDL–tiedoston sisällön.



Kuva 9 Sovelluspalvelu lisättynä projektiin

5.2 Tulosten tallennus

Tuloksen tallennus tapahtui SoapClient-instanssilla. Uuden instanssin luonnin jälkeen tehtiin verkkoTulos-muuttuja, joka oli sovelluspalvelun tallennusfunktion tietosopimus Tulos-tietotyyppiä. SOAP-protokollaa käytettäessä Tu-

los-tietotyyppiä ei tarvinnut erikseen luoda, vaan tietotyypin pystyi hakemaan suoraan palveluviittauksista koodiesimerkissä 11 olevalla tavalla. Verkkotulokseen sijoitetaan kyseisen pelikierroksen pelitunnus ja pisteet. Ennen tallennusta tarkistettiin vielä, että kännykkä on yhteydessä verkkoon. Tallennus tapahtui lisäämällä client-instanssille TallennaTulosCompleted-käsittelijä ja kutsumalla TallennaTulosAsync-metodia. Parametrina toimi metodin vaatima Tulos-tietotyypin instanssi.

Koodiesimerkki 11 Tuloksen tallennus

```
Dim client As New SoapClient
Dim verkkoTulos = New mobiilipeli.wsTulospalvelu.Tulos()
verkkoTulos.Pisteet = tbTulos.Text
verkkoTulos.Tunnus = tbNimi.Text

''''''''''''''NETTIYHTEYS''''''''''''''
If NetworkInterface.GetIsNetworkAvailable() = True Then

    AddHandler client.TallennaTulosCompleted, AddressOf TallennusValmis
    client.TallennaTulosAsync(verkkoTulos)

End If
```

5.3 Tulosten luku

Tulosten luku tietokannasta toteutettiin REST-tekniikan operaatiosopimuksia hyväksikäyttäen.

Koodiesimerkki 12 HttpWebRequest-kutsut

```
''''''''''''''NETTIYHTEYS''''''''''''''
If NetworkInterface.GetIsNetworkAvailable() Then

    http = HttpWebRequest.Create(New Uri("[redacted]/haetop10/"))
    http2 = HttpWebRequest.Create(New Uri("[redacted]/haesija/" & intPisteet))

    http.BeginGetResponse(New AsyncCallback(AddressOf readcallback), http)
    If intPisteet > 0 Then
        http2.BeginGetResponse(New AsyncCallback(AddressOf readcallback2), http2)
    End If

End If
```

Koodiesimerkissä 12 aloitetaan kaksi HttpWebRequest-kutsua. Ensimmäisellä haettiin kymmenen parasta tulosta tietokannasta ja toisella oman tuloksen sijoitus. HttpWebRequest-kutsu ottaa parametrina URI-instanssin, jossa osoite on tarvittavan sovelluspalvelun operaatiosopimuksen määrittelyssä asetettu osoite. BeginGetResponse aloittaa kutsun. Koodiesimerkissä 13 esitetään takaisinkutsufunktioiden toiminta. Funktiot lukevat sovelluspalvelulta tulevan XML-muotoisen datan ja jäsentää tarvittavat tiedot siitä.

HttpWebRequest-kutsun ollessa käynnissä kaikki käyttöliittymäpäivitykset täytyy ajaa käyttäen Me.Dispacher.BeginInvoke-funktiota. Tämä johtuu siitä, että takaisinkutsu jäädyttää käyttöliittymäsäikeen, mistä seuraa ettei käyttöliittymää voida kutsun aikana päivittää. BeginInvoke mahdollistaa käyttöliittymän päivityksen takaisinkutsun aikana.

Tulokset tallennettiin verkkoscores muuttujaan. Tämä muuttuja on ObservableCollection -kokoelma, joka on Tulos-luokan tietotyyppiä. Tulos-luokan toteutus näkyy koodiesimerkissä 14. Toteutuksessa on määritelty Tulos-luokalle järjestystä, tunnusta ja pistemäärää vastaavat ominaisuudet. Muodostimia luotiin kaksi. Toisessa järjestystä ei tarvitse antaa.

Koodiesimerkki 13 Takaisinkutsufunktiot

```
Private Sub readcallback(callbackResult As IAsyncResult)
    Dim myrequ As HttpWebRequest = callbackResult.AsyncState

    If Not myrequ Is Nothing Then
        Using myResp As HttpWebResponse = myrequ.EndGetResponse(callbackResult)
            Using strred As StreamReader = New StreamReader(myResp.GetResponseStream())
                Dim str As String = strred.ReadToEnd()
                Me.Dispatcher.BeginInvoke(Function() muutateksti(str))
            End Using
            myResp.GetResponseStream.Dispose()
        End Using
    End If


End Sub
```

```
Private Function muutateksti(ByVal str As String) As System.Action

    verkkoscores.Clear()
    Dim reader As XElement = XElement.Parse(str)
    Dim resultElement As XElement = reader.Descendants(XName.Get("NewDataSet")).Single()

    Dim index As Integer = 1
    For Each element As XElement In resultElement.Elements
        verkkoscores.Add(New Tulos(index, _
                                   element.Descendants(XName.Get("T_tunnus")).Value, _
                                   element.Descendants(XName.Get("T_pistemaara")).Value))
        index += 1
    Next
    index = 0

    Return Nothing
End Function
```

```
Private Sub readcallback2(callbackResult As IAsyncResult)
    Dim myrequ As HttpWebRequest = callbackResult.AsyncState

    If Not myrequ Is Nothing Then
        Using myResp As HttpWebResponse = myrequ.EndGetResponse(callbackResult)
            Using strred As StreamReader = New StreamReader(myResp.GetResponseStream())
                Dim str As String = strred.ReadToEnd()
                Me.Dispatcher.BeginInvoke(Function() muutateksti2(str))
            End Using
            myResp.GetResponseStream.Dispose()
        End Using
    End If


End Sub
```

```
Private Function muutateksti2(str As String) As System.Action

    tbVerkkojarjestys.Text = XElement.Parse(str).Value & "."

    Return Nothing
End Function
```

Koodiesimerkki 14 Tulosluokka

```
Imports System.Runtime.Serialization

<DataContract(Namespace:="")>
Public Class Tulos
    Public Property Jarjestys() As Integer
        Get
            Return m_Jarjestys
        End Get
        Set(value As Integer)
            m_Jarjestys = value
        End Set
    End Property
    Private m_Jarjestys As Integer

    <DataMember()>
    Public Property Tunnus() As String
        Get
            Return m_Tunnus
        End Get
        Set(value As String)
            m_Tunnus = value
        End Set
    End Property
    Private m_Tunnus As String

    <DataMember()>
    Public Property Pisteet() As Integer
        Get
            Return m_Pisteet
        End Get
        Set(value As Integer)
            m_Pisteet = value
        End Set
    End Property
    Private m_Pisteet As Integer

    Public Sub New(nimi As String, pisteet As Integer)
        Me.Tunnus = nimi
        Me.Pisteet = pisteet
    End Sub

    Public Sub New(jarjestys As Integer, nimi As String, pisteet As Integer)
        Me.Jarjestys = jarjestys
        Me.Tunnus = nimi
        Me.Pisteet = pisteet
    End Sub

End Class
```

5.4 Tulosten liittäminen ListBox-komponenttiin

Tulokset järjestettiin ListBox-komponenttiin kolmeen sarakkeeseen, joista ensimmäisessä on tuloksen järjestys, toisessa tuloksen tekijän nimi ja kolmannessa pistemäärä. Järjestyksen esityksessä käytettiin muunninta, jolla saatiin lisättyä numeron perään piste. Tiedot sidottiin listaan käyttämällä tulos luokassa määriteltyjä nimiä. Näin ollen, kun asetettiin ListBox-komponentin ItemsSource viittaamaan verkkoscores kokoelmaan tulevat tulokset näkymään listalla (listboxVerkkoTulokset.ItemsSource = verkkoscores). Kuvassa 24 näytetään ListBox-komponentin toteutus .xaml tiedostossa.

Koodiesimerkki 15 ListBox-komponentin toteutus

```
<ListBox x:Name="listboxVerkkoTulokset" Background="Beige" Foreground="Black" Margin="0,-20,0,0">
    <ListBox.Resources>
        <src:NumberConverter x:Name="NumberConverter2" />
    </ListBox.Resources>
    <ListBox.ItemContainerStyle>
        <Style TargetType="ListBoxItem">
            <Setter Property="HorizontalContentAlignment" Value="Stretch"/>
        </Style>
    </ListBox.ItemContainerStyle>
    <ListBox.ItemTemplate>
        <DataTemplate>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="35"/>
                    <ColumnDefinition />
                    <ColumnDefinition Width="230"/>
                </Grid.ColumnDefinitions>
                <TextBlock Grid.Column="0" HorizontalAlignment="Center" FontSize="23" x:Name="tbJarjestys"
                           Text="{Binding Jarjestys, Converter={StaticResource NumberConverter2}}"/>
                <TextBlock Grid.Column="1" FontSize="23" x:Name="tbNimi" Text="{Binding Tunnus}"/>
                <TextBlock Grid.Column="2" HorizontalAlignment="Right" Margin="0,0,10,0" FontSize="23"
                           x:Name="tbPisteet" Text="{Binding Pisteet}"/>
            </Grid>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

Kuvassa 10 on Windows Phone –emulaattorista kaapattu kuva, jossa tulokset on ladattu puhelimeen tietokannasta käyttäen sovelluspalvelua.

Kuva 10 ListBox-komponenttiin ladatut tulokset

6 PÄÄTELMÄT

Www-sovelluspalveluiden käyttö mobiilisovelluksissa on nykyään hyvin perusteltua. Älypuhelimien sovellusvalikoimien lisääntyessä yhä monimutkaisempia sovelluskokonaisuuksia pystytään luomaan. Sovelluspalvelut tuovat sovelluksiin paljon uusia mahdollisuuksia. Tärkeimmät www-sovelluspalveluiden hyödyt kuten alustariippumattomuus, ohjelmointikieliriippumattomuus ja sovellusten integrointi ovat tuoneet verkkopalvelut kaikkien ohjelmistosuunnittelijoiden tietoisuuteen.

WCF-sovelluspalvelu tuo sovelluskehittäjille mahdollisuuden luoda niin pienen kuin suuren mittakaavan www-sovelluspalvelukokonaisuuksia. WCF Web Service -palvelun perusteiden opettelu voi olla aikaa vievää, koska käsitteenä WCF on hyvin laaja, mutta alkuun pääsee nopeasti.

Omalta osalta työn toteutus sujui ilman suurempia ongelmia. Haasteita toi kuitenkin sovelluspalvelupuolen toteutus, sillä aiempaa kokemusta WCF-sovelluspalveluista ei ollut. Eniten aikaa vei erilaisten WCF-sovelluspalveluihin liittyvien käsitteiden ja standardien opettelu. Lopulta saatiin kuitenkin toimiva kokonaisuus, jonka jatkokehitys on WCF-palveluista tietävälle hyvin helppoa.

LÄHTEET

[1] Web Service 2012. Wikipedia. Viitattu 20.9.2012 http://fi.wikipedia.org/wiki/Web_service.

[2] Geetanjali, A. & Geetanjali, K. 2002. Building web services with XML. USA: Premier Press Incorporated.

[3] Abeysinghe, S. 2008. RESTful PHP Web Services. Iso-Britannia: Packt Publishing Ltd.

[4] Balani, N. & Hathi, R. 2009. Apache CXF Web Service Development: Develop and Deploy SOAP and RESTful Web Services. Iso-Britannia: Packt Publishing Ltd.

[5] Tost, A. 2003. Web services programming tips and tricks: Using SOAP headers with JAX-RPC, Viitattu 15.10.2012 http://www.ibm.com/developerworks/webservices/library/ws-tipjax1.html.

[6] Soap Syntax 2012. W3Schools. Viitattu 25.10.2012 http://www.w3schools.com/soap/soap_syntax.asp.

[7] UDDI 2012. Viitattu 30.11.2012 http://uddi.xml.org/uddi-101.

[8] Claeys, K.; Cibraro, P. & Grabner, J. 2010. Professional WCF 4: Windows Communication Foundation with .NET 4. USA: Wrox.

[9] Configuring System-Provided Bindings 2012. Microsoft Developer Network. Viitattu 20.12.2012 http://msdn.microsoft.com/en-us/library/ms731092.aspx.

[10] Choosing the right wcf binding. Lowy, J. Viitattu 1.11.2012 http://weblogs.asp.net/spano/archive/2007/10/02/choosing-the-right-wcf-binding.aspx.

[11] Endpoints: Addresses, Bindings, and Contracts. Microsoft Developer Network. Viitattu 19.11.2012 http://msdn.microsoft.com/en-us/library/ms733107.aspx.

[12] Designing Service Contracts. Microsoft Developer Network. Viitattu 20.10.2012 http://msdn.microsoft.com/en-us/library/ms733070.aspx.

[13] How to: Host a WCF Service in a Managed Application. Microsoft Developer Network. Viitattu 10.12.2012 http://msdn.microsoft.com/en-us/library/ms731758.aspx.

[14] How to: Host a WCF Service in a Managed Windows Service. Microsoft Developer Network. Viitattu 15.11.2012 http://msdn.microsoft.com/en-us/library/ms733069.aspx.

[15] The Official Microsoft IIS site. Viitattu 27.11.2012 http://www.iis.net/

[16] How to: Host a WCF Service in IIS. Microsoft Developer Network. Viitattu 1.12.2012 http://msdn.microsoft.com/en-us/library/ms733766.aspx.

[17] How to: Host a WCF Service in WAS. Microsoft Developer Network. Viitattu 20.12.2012 http://msdn.microsoft.com/en-us/library/ms733109.aspx.